Remarks

Status of application

Claims 1-45 were examined and stand rejected in view of prior art. The claims have been amended to further clarify Applicant's invention. Reexamination and reconsideration are respectfully requested.

The invention

A database system providing methodology for extended memory support is described. In one embodiment, for example, Applicant's invention provides a method for extended memory support in a database system having a primary cache, where the method comprises steps of: creating a secondary cache in memory available to the database system; mapping a virtual address range to at least a portion of the secondary cache; when the primary cache is full, replacing pages from the primary cache using the secondary cache; in response to a request for a particular page, searching for the particular page in the secondary cache if the particular page is not found in the primary cache; if the particular page is found in the secondary cache, determining a virtual address in the secondary cache where the particular page resides based on the mapping; and swapping the particular page found in the secondary cache with a page in the primary cache, so as to replace a page in the primary cache with the particular page from the secondary cache.

General

Dependent claims 22 and 23 are rejected on the basis that they are product claims, while their independent claim 1 is a method claim. Additionally, claim 23 was rejected under Section 101 as being directed to nonstatutory subject matter (i.e., a computer program per se). The claims have been amended such that the claim language is now directed to the subject matter of a method (i.e., statutory subject matter, with both being method claims like their parent, independent claim 1).

Prior art rejections

A. Section 102(e) rejection: Alsup

Claims 1-3, 5-26, and 28-45 stand rejected under 35 U.S.C. 102(e) as being

anticipated by Alsup (US No.: 2004/0103251 A1). Here, the Examiner likens Applicant's claimed invention to a microprocessor's L2 (Level 2) secondary cache. For the reasons stated below, Applicant's claimed invention may be distinguished on a variety of grounds.

At the outset, it is important to recognize that there are a variety of existing caching schemes, including a variety of secondary caching schemes. Both a microprocessor's on-board L2 cache mechanism and Applicant's claimed invention fall within the general subject matter of secondary caching mechanisms, and of necessity will overlap somewhat as to general features. However, Applicant makes no claim to have invented the notion of a secondary cache. Instead, Applicant's claimed invention pertains to a software-implemented cache in system memory that is specifically designed for improving performance of database systems.

Alsup describes the L2 (secondary) cache mechanism that is implemented on board microprocessors (i.e., cache is physically located on the microprocessor chip). With the continued wafer and die size growth, defect density reduction, and increased transistor density of modern-day microprocessors, vendors such as Intel and AMD have employed increasingly available die "real estate" to implement increasingly larger on-board microprocessor L1 and L2 caches. An L2 cache is of course a dedicated (i.e., hard wired) cache that directly supports the operations of the primary (L1 or Level 1) cache, which also resides directly on-board the microprocessor. As such, both the L1 and L2 caches are very specific features of microprocessors, and of course require direct support by the microprocessor. Not only is the L2 cache physically "burned" (i.e., placed) onto the microprocessor chip, the microprocessor includes specific firmware (i.e., hard coded instructions) burned on to the chip to support operation of the L2 cache. Clearly, a microprocessor-based L2 cache is a very specific hardware cache solution for supporting operations of the microprocessor: working in conjunction with the microprocessor L1 cache, the L2 cache improves the loading of executable code from system memory so that the microprocessor may execute faster.

Applicant's claimed invention is not directed to any caching mechanism for use on a processor. Instead, Applicant's claimed invention is a software-implemented (i.e., media-stored instructions for execution by a general-purpose microprocessor) cache

mechanism that uses ordinary system memory (i.e., memory located external to the microprocessor). Applicant's claimed invention uses a memory mapped file (i.e., an operating system-supported feature) to create a secondary cache in system memory, for improving the caching of database pages (i.e., specific runtime database objects). Significantly, Applicant's approach extends available system memory (e.g., from 4GB to 64GB) without any specific hardware support.

Had Applicant merely invented a software-implemented replacement for a microprocessor's L2 cache mechanism, it is respectfully submitted that that would have been a novel and perhaps useful invention. However, Applicant's invention is not directed to caching byte blocks containing code for execution by a processor's execution unit. Instead, at a core level, Applicant's claimed invention is a software-based solution directed at caching software objects -- objects that are fundamentally different than the bytes of code instruction transferred from L2 cache to L1 cache (and then to the microprocessor's execution unit) in Alsup's system. In order to better understand this distinction, it is instructive to examine the runtime memory management needs of a database server.

A database server, during operation, will employ a cache in system memory for caching database objects (e.g., data pages for tables, indexes, and the like). It is advantageous to cache this type of data in memory so that the data does not have to be brought into memory from an external source (e.g., disk) each time an operation on the data is to be performed. For example, in response to a request to fetch database data (e.g., pursuant to an SQL query), a database server may find it advantageous to cache a particular index.

The foregoing "primary" cache is a software-based cache used for caching runtime program objects. It is implemented in general-purpose system memory in response to software (i.e., media-stored instructions executable on a general purpose microprocessor) loaded from disk, during operation of the software program (e.g., relational database software), and is used to cache runtime program objects (i.e., objects discernible at the source code level, such as database tables, database indexes, etc.). Note that this software-based cache, which is operating at runtime to cache program <u>data</u> objects, operates concurrently and independently of any processor-based caching

mechanism. A processor-based cache, in contrast, operates on raw memory units (byte block), typically functioning to load code (i.e., microprocessor-executable instructions) for execution by the processor's execution unit. Importantly, there is no way to use the processor-based cache to bind frequently-used runtime data objects (e.g., database tables or indexes), so that those data objects are readily accessible to the database software. Instead, the processor-based cache improves the execution of the code that comprises the database software, but does little or nothing for runtime data objects.

The primary cache that is available to an application (e.g., a database server) is often limited, and thus Applicant's invention provides a specific improvement. To the extent that the data used by the application is not available in the primary cache (especially, due to a cache size constrained to less than 4GB on 32-bit machines), it needs to be brought in from another source, namely from disk or other external storage. However, disk reads/writes are expensive and adversely impact performance of the application. Applicant's invention solves this problem by providing extended memory support though the creation of a secondary cache that acts like a swap space for the primary cache.

An L2 cache mechanism, such as described by Alsup, does not address the above identified problem that is solved by Applicant's invention. Specifically, in certain architectural scenarios, such as Linux running on a 32-bit Intel microprocessor, it is desirable to support a very large address space (e.g., 64GB) even though the platform itself only supports a 4GB memory space. Applicant's invention provides an approach for mapping to extended memory on various devices so that an application can bring data in from various sources without needing to know or be concerned about the underlying location of this data. In the particular instance of Applicant's claimed invention, a secondary cache is created in system memory using a memory mapped file. These features, for which there is no analogue found in processor-based caches, are especially important for memory intensive applications, such as large database systems, that benefit from the ability to address more memory than can be supported in the address space of a given platform (e.g., a 32-bit platform).

The use of a memory mapped file space to create a secondary cache from which the system replaces the pages from the primary cache is gently advantageous: it enables

the database system to address a portion of memory mapped file whose size can be as large as 64G (e.g., in an embodiment operating on a 32-bit platform). While searching for a given page in cache, if that page is cached in the secondary cache, the system maps a virtual address window to that portion of the memory mapped secondary cache where the page resides and copies it to the primary cache. By doing so, the system avoids disk reads while making effective use of memory mapping and shared memory features of the operating system.

Although Applicant's original claims are believed to distinguish over the cited art, the claims have nevertheless been amended to expedite prosecution of the present application. For example, independent claim 1 has been amended to now include claim language (shown in amended form):

> using a memory mapped file, creating a secondary cache in system memory available to the database system;

As shown, the amendment highlights the above-described features that distinguish Applicant's invention. The amendment prevents any interpretation of the claims from reading on a processor-based cache mechanism, such as described by Alsup.

Further, Applicant's claims have been amended to further clarify the nature of the objects being cached, as shown by the following amended claim language (shown in amended form):

> when the primary cache is full, replacing database pages from the primary cache using the secondary cache;
>
> in response to a request for a particular database page, searching for the database particular page in the secondary cache if the particular database page is not found in the primary cache;
>
> if the particular database page is found in the secondary cache, determining a virtual address in the secondary cache where the particular database page resides based on the mapping; and
>
> swapping the particular database page found in the secondary

cache with a database page in the primary cache, so as to replace a database page in the primary cache with the particular database page from the secondary cache.

Significantly, the caching mechanism in Applicant's invention caches software objects, such as database pages for a database system. The above amendment to the claims makes this distinction explicit. (Applicant's other independent claim, claim 24, already includes claim language indicating that data pages are the software objects being cached.)

For the reasons stated above, it is respectfully submitted that the amended claims set forth a patentable advance in the caching art. In view of the foregoing clarifying amendments made to the claims (as well as remarks made above), it is believed that the claims distinguish over the cited art in any rejection under Section 102(e) is overcome.

B. Section 103(a) rejection: Alsup in view of Austin

Claims 4 and 27 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Alsup (above) in view of Austin et al., (US No: 20030162544 A1, hereinafter "Austin"). Here, the Examiner repeats the rejection of Alsup above, but adds Austin for the proposition that it teaches use of the use of a Linux operating system. The claims are believed to be allowable for at least the reasons cited above pertaining to the Section 102 rejection based on Alsup. Nothing in Austin cures the deficiencies of Alsup.

The claims are believed to be allowable for the following additional reasons. Claim 4, when read in conjunction with intervening claim 3 and base claim 1, establishes that Applicant's claimed secondary cache is created using Linux operating system's shared memory file system. This is very different than an on-board processor cache. Significantly, the claim (when read in conjunction with claims 1 and 3) does not merely add the use of the Linux operating system (e.g., taught by Austin), but instead specifies a software-implemented secondary cache created in system memory using the Linux shared memory file system (as well as using Linux memory mapped file feature). Rejected claim 27 sets forth similar limitations.

These claim limitations are not taught or suggested by the combination of Alsup

with Austin. And in particular, the Examiner has left unaddressed implementation of a software-based secondary cache using an operating system's shared memory file system. Accordingly, the claims are believed to distinguish over the combined references (especially, in view of clarifying amendments made to the base claims), and that any rejection under Section 103 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: February 9, 2007

/John A. Smart/

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX